(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 813 139 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
17.12.1997 Bulletin 1997/51

(51) Int. Cl.[6]: **G06F 3/033**

(21) Application number: **97304137.9**

(22) Date of filing: **13.06.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **14.06.1996 US 19782 P**
31.01.1997 US 792971

(71) Applicant: **THE FOXBORO COMPANY**
Foxboro, MA 02035 (US)

(72) Inventor: **Calder, Dale E.**
Mansfield, Massachusetts 02048 (US)

(74) Representative:
Downing, Michael Philip et al
Fry Heath & Spence,
The Old College,
53 High Street
Horley, Surrey RH6 7BN (GB)

(54) Dynamic property sheet system

(57) Methods and apparatus for the concurrent display of two or more property pages of a Graphical User Interface (GUI) are disclosed. A user can select a property page displayed by a first property sheet system and create a second property sheet system that displays the selected property page at second, user selected display position. Alternatively, the user can place, or dock, the selected property page into an existing property sheet system for inclusion therein and display thereby. The first property sheet system displays, at a first display position, any of the property pages remaining therein. Typically, the selected property page is removed from the first property sheet system and the first property sheet system is destroyed if it contains no other property pages. The invention allows a user of a GUI to concurrently display and interact with multiple property pages of multiple property sheet systems for optimizing the exchange of information with the GUI.

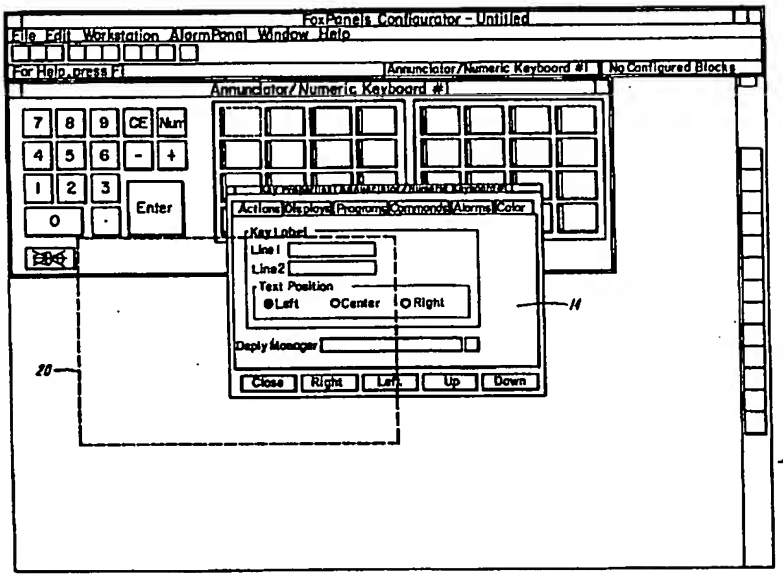


FIG. 3

EP 0 813 139 A2

Description

Field of the Invention

The present invention relates to graphical user interfaces, and particularly to property sheet systems having property pages viewable on a graphical user interface.

Background

A graphical user interface (GUI) is an indispensable feature of most modern computer systems. A GUI allows a user to interact intuitively with an application program running on the computer by allowing the user to manipulate pictorial representations of objects. Many of the display components of a typical GUI, such as title bars, menu bars, icons, multiple windows, scroll bars, pop up menus and dialog boxes, are now standard features. A modern GUI represents a major improvement over the text-based entry systems of just a few years ago.

Typical dialog boxes of a GUI system allow a user to exchange information with an application program. Some dialog boxes simply require a user to confirm an operation before it is executed by the computer; other dialog boxes may ask for user selected input, such as choice of a disk drive, directory, file name, file type, network path, etc. Dialog boxes typically comprise components such as text boxes, list boxes, option buttons, and command buttons.

If a large amount of related information is to be exchanged between a user and the computer a GUI may include a special type of dialog box known as a property sheet system. A property sheet system, for example, can have a display that includes several tabs which represent different dialog boxes, known as a property pages. The user can select a tab to display the particular property page associated with that tab. Once selected, the user can enter information, confirm an operation, or make other appropriate option choices.

Available property sheet systems can be frustrating to use. For example, to configure optimally an application program, a user often exchanges information with the program by using several modeless property pages. Often the configuration options that result in the desired performance of the application are not immediately obvious and are arrived at by some trial and error. Unfortunately, because the property sheet system displays only one property page at a time on a monitor, a user usually selects appropriate tabs to switch between multiple property pages, e.g. property pages are relegated to the background Furthermore, the user is typically required to remember the particular settings on the nondisplayed property pages. This procedure of continually switching between property pages to configure a system is overly laborious and time consuming.

Accordingly, an object of the invention is to provide a property sheet system that allows a user to easily and efficiently exchange information with an application program.

A further object of the invention is to provide a property sheet system that reduces the need of a user to choose continually between displayed and nondisplayed property pages when the exchange of information requires the use of more than one property page.

Other general and specific objects of the present invention will be apparent and evident from the accompanying drawings and the following description.

Summary of the Invention

The present invention relates to, broadly, the creation of a second property sheet system from a first property sheet system. The first property sheet system includes one or more property pages, which are displayable on a display monitor at a first display position. The first property sheet system is often part of an existing application program. For example, a word processor program on which this application is being typed has a property sheet system for setting up printers. This property sheet system includes "tabbed" property pages entitled "setup", "paper", "destination", "print quality", and "fonts". According to the invention, a property page is selected is response to an input signal from a user-operated pointing and selecting device, such as a mouse. In response to a second signal generated by the user, typically via the mouse, the selected property page is included in, and displayed by, a second property sheet system. The user may thus display and interact with two property pages simultaneously: the selected property page, which is displayed by the second property sheet system, and any one of the property pages remaining in the first property sheet system, which is displayed by the first property sheet system.

One technique by which a user may select a property sheet is by "tearing," that is, selecting a tab representing a property page of the first property sheet system. The property page "torn" or selected from the first property sheet sys-

tem is displayed at a second location, determined by the user, on the display screen. Typically, a user employs "drag and drop" operation to select a property page, create a second property sheet system, and to display the torn property page at a user-selected display location.

As noted, creating more than one property sheet system allows a user to view and exchange data with an application program using two or more property pages that are viewed simultaneously. The user need not continually switch between displayed and non displayed property pages, nor need he or she commit to memory the contents of one or more property pages. Note that as used herein, the term property page or sheet is intended to include a modal or modeless secondary window that displays user-accessible properties of an object. A property page is said to be included in a property sheet system and can be displayed in a variety of formats to best allow a user to exchange information with a computer or with an application program running on a computer.

Additionally, property pages can be exchanged between property sheet systems, whether these systems are created according to the invention or exist as part of a separate application program, such as a word processor program. A property page "torn" from a property sheet system is typically displayed as a geometric object or other symbol during the "drag" portion of the "drag and drop" operation. A user can place the selected, or "torn," property page in an existing property sheet system for inclusion therein and display thereby, rather than in a new property sheet system created to include the selected page. Typically, the user places the symbol representing the "torn" page in close proximity to a property page displayed by the property sheet system into which it is desired to drop the "torn" page. A docking input signal, such as release of a mouse button, causes the selected property page to "drop", or "dock" into the chosen property sheet system. Once docked in the chosen system, the property page may be used by the user just as when it was in the previous property sheet system.

According to one aspect, the invention includes a processor; a memory element; a display monitor; a user input device; a selection element for selecting a property page of a first property sheet system responsive to a signal generated by a user input device, such as, for example, mouse, trackball, or keyboard; and a second program element for creating a second property sheet system for including the selected property page and for displaying the selected property page at a second display position on the display monitor.

According to another aspect, the invention includes a removal element for removing from the first property sheet system the property page selected by the selection element such that the selected property page is not displayable by the first property sheet system.

In yet another aspect, the invention includes a destruction element for destroying a property sheet system when system does not include any property pages, such as when the last page in the system has been removed by the removal element.

According to another aspect of the invention the invention further includes a docking element, for, in response to a signal generated by a typical user input device, placing a property page selected by the selection means into a selected property sheet system for inclusion therein and display thereby.

In yet another aspect, the invention incorporates a positioning program element, responsive to input signal generated by the user input device, for selecting the second display position at which the second property sheet system displays property sheets included therein.

In a different variation, the positioning element varies the display position of a symbol in response to a user input. The second program element, responsive to the user input, displays the selected property page at the second position. The positioning element also allows for placing the symbol in substantial registration with a displayed property page of a property sheet system such that the docking element, responsive to a user input signal, such as release of a mouse button, places the selected property page into the property sheet system with which the symbol was in substantial registration. The symbol can be the dotted outline of a box, and the box can be the approximate size of a displayed property page.

To aid the user in docking a selected property page into an existing property sheet system, the invention can include an element for displaying on the display monitor an icon having a second display state for indicating to a user when the symbol or geometric object is in substantial registration with a property page of a property sheet system. For example, the icon second display state can comprise a display of a two puzzle pieces that are mated, and the icon can have a first display state comprising two puzzle pieces that are not mated, for indicating a when a symbol or geometric figure is not properly registered, i.e., the selected property page will not be docked by the docking means into an existing property sheet system.

According to another feature of the invention, in addition to a positioning element responsive to user input for varying a position of symbol displayed on the display monitor, a a proximity program element is included. The proximity element determines a parameter responsive to the distance on the display monitor between the symbol and the position on the display monitor of a displayed property page of a property sheet system. When the parameter is less than a first amount, the docking means, responsive to a first user input signal generated by the user input device, places the property page selected by the selection element into the property sheet system for inclusion therein and display thereby. When the parameter varies from the selected value by a second selected amount, the second program means, responsive to a second input signal, places the property page in a newly created property sheet system and displays the

selected property page at a position on the display monitor substantially determined by the display position of the symbol. For example, the displayed property page can be centered about the position of the symbol. The first and second input signals can be responsive to the release of a mouse button as part of the "drag and drop" procedure.

Of course, the invention pertains not only to a computer programmed in the manner described above to perform the transformation and operations described above, but to a machine readable data storage medium including the above-described elements. The invention is also understood to comprise methods for performing the above operations.

It will thus be seen that certain changes may be made in the above constructions without departing from the scope of the invention. It is intended that all matter contained in the above and following description or shown in the accompanying drawings be interpreted as illustrative and not in a limiting sense.

It is also to be understood that the following claims are to cover all generic and specific features of the invention described herein, and all statements of the scope of the invention which, as a matter of language, might be said to fall therebetween.

Brief Description of the Drawings

The foregoing and other objects, features and aspects of the invention will be apparent from the following description and apparent from the accompanying drawings, in which like reference characters refer to the same parts throughout the different views. The drawings illustrate principles of the invention and, although not to scale, shown relative dimensions.

FIGURE 1 illustrates a typical computer system according to the invention.

FIGURE 2 illustrates a typical property page of a property sheet system for configuring, for example, the keyboard display of an application program. A user accesses each property page of the property sheet system by selecting the appropriate tab displayed at the top of the property page window display.

FIGURE 3 illustrates an intermediate display presented as part of the "drag and drop" procedure for creating a new property sheet system including a property page "torn" from the first property sheet system of FIGURE 2.

FIGURE 4 illustrates a second property sheet system including the property page "torn" from the first property sheet system illustrated in FIGURE 2 and which is displayed simultaneously therewith.

FIGURES 5A and 5B depict the selected display states of a dynamic icon for aiding a user in "docking" a property page removed from one of the property systems for placement within another sheet system.

FIGURE 6 is a flow chart diagram illustrating the steps involved in removing a property page from a property sheet system and placing it in a second property sheet system, as depicted in FIGURES 2-4.

FIGURE 7 is a schematic illustration block diagram of the software components utilized in accordance with the teachings of the present invention;

FIGURE 8 illustrates the object classes, and their relationships, in accordance with the teachings of the present invention.

FIGURES 9A-9C illustrate parent-child relationships of particular object classes depicted in FIGURE 8 and used in the software coding according to the teachings of the invention.

FIGURE 10 is a hierarchical schematic illustrating the derivation of key property pages from a property page base class according to the teachings of the present invention.

Description of Illustrated Embodiments

The removing or tearing of property pages from a property sheet system, in accordance with the teachings of the present invention, extends the user interface of standard Windows/Windows 95/Windows NT property pages to enable a user to change values on separate property pages simultaneously without changing visual context. The user can grab the active property page and pull/tear it away from the property sheet system and place it in an existing or newly created property sheet system. However, one of ordinary skill in the art, in accordance with the teachings herein, understands the invention to be applicable to GUI's in other operating environments, such as, for example, X-Windows, Motif, and IBM OS/2.

The practice of the invention includes a computer, such as the computer 2 illustrated in FIGURE 1. The computer 2 typically includes input-output hardware 3, a central processor 4, random access memory 5, disk storage 6, a keyboard 7, pointing device, such as a mouse 8 and a display monitor 10. The Input/output hardware 4 connects the computer 2 to, for example, a network.

FIGURE 2 illustrates a typical computer display screen display 12 of a property page 14 of a property sheet system. The property page 14 in FIGURE 2 is used for configuring the keyboard display 16 of an application program that monitors and controls industrial processes. One such industrial process control system is the I/A SERIES® System available from The Foxboro Company, of Foxboro, Massachusetts, USA. However, an industrial process monitoring system is merely one example of the type of environment in which property pages are typically very useful; the present invention is useful in many computer application programs, such as word processors, spreadsheets, communications pack-

ages, web browsers, etc. However, in the particular embodiment illustrated here, the property pages are used to configure a keypad annunciator of an industrial process control system, such as the keypad annunciator 16 in FIGURE 2. Many of the keys of a keypad annunciator 16 are configurable, via property pages, such that a given individual key is associated with a particular alarm state of the industrial process control system; occurrence of that alarm state causes the key to change appearance. The key is also configurable such that selecting the key with the mouse can initiate selected corrective action or display a selected process graphic.

To "tear" a page from a property sheet system, the user positions a cursor over the body (i.e., generally anywhere other than an edit field or the tab representing the property page 14) of property page 14 and provides an input signal, typically by depressing a button on the mouse. During this tearing, or selection, process the user is presented with an outline of the property page while the mouse is pressed, and the outline can be positioned by "dragging" it anywhere on the display screen. FIGURE 3 illustrates the outline 20 of property page 14 that is in the process of being torn from its property sheet system. Once the desired position is found, releasing the mouse button signals the application to create a new property sheet dialog system and populate it with the torn-away property page. Typically, the original property page 14 is removed from its original container, e.g., the first property sheet system, and is only available in the second property sheet system. The user can view and change values using different property pages in one visual context, that is, to view them simultaneously positioned above the desktop 13. FIGURE 4 illustrates the simultaneous display of two property pages, the torn page 14, displayed in the display window 24 of the newly created property sheet system, and property page 22, now displayed in the original property sheet system display window 15, and corresponding to the tab 25 labeled "displays." The tab 18 labeled "actions" in FIGURE 2 is removed from property sheet display window 15 and now appears in display window 24 in the newly created property sheet system, as illustrated in FIGURE 4. The property page 14 and the property page 22 are now displayed concurrently, and the user can exchange data with either of them. Typically, as the user interacts with one of the displayed property pages, the other is updated, as necessary, to reflect the data exchange with the other. For example, if the display window 24 of the newly created property sheet system is active, that is, the user is exchanging data with the property page 14, the property page 22, displayed in the non-active window 15 of the original property sheet system, is updated, as necessary, to reflect the any data input by the user to the property page 14 displayed in window 24.

In addition to being able to create a new property sheet system for displaying the selected property page, the user can place the selected property page into an existing property sheet system. The user attaches, or docks, a "torn" property page into an existing property sheet system by dragging the outline 20 over the top of a property page displayed by that system. Releasing the mouse button when the outline is so positioned docks the property page into the property sheet system. As discussed immediately above, a displayed property page in a non active window is updated, as necessary, to reflect any input by a user to a displayed property page in an active window. Using the docking procedure, the property page 14 in FIGURE 4 can be selected and placed back into in the original property sheet system for redisplay in display window 15.

A visual cue, such as a dynamic icon 28 in FIGURES 5A and 5B, is provided to assist in the tear-away/docking operation. During the dragging process, the user is provided with a symbol such as dynamic icon 28 that shows two puzzle pieces. When the drag box is over a dock-site, i.e., over an existing property sheet system that can accept the property page being dragged, the puzzle pieces join and appear as a second display state 32, thus signifying to the user that the dragged page can be docked e.g., accepted into the that system. FIGURE 5A illustrates the dynamic icon in the first display state 30 wherein the puzzle pieces are separated, indicating that the drag box or geometric shape 20 is not properly positioned for a property page to be docked.

The invention uses the logic depicted in the operational flow chart illustrated in Figure 6. Starting at decision box 34, pressing a mouse key when the cursor is within a property page "tears" the property page and leads along the "yes" branch of box 34 to box 36. As the mouse is dragged, that is, moved with the mouse button pressed, an outline of the property page is drawn on the screen. This process will continue until the user releases the mouse button. Releasing the mouse button leads from along the "yes" branch of box 36 to decision box 38, which determines whether the torn property page can be docked, that is, whether the drag box 20 is positioned over a property page of an existing property sheet system. If docking is possible, the torn page is docked into the "target" system, as in box 42; if the page cannot be docked, a new property sheet system is created for inclusion of the torn property page, as indicated by box 40. Finally, in box 44, the torn page is removed from the original property sheet system, and if no other pages remain in the original, or source, property sheet system, the system is deleted, or destroyed. In the remainder of the specification, the object classes and coding to perform the above-described operations are described.

The present invention uses techniques of object-oriented programming. In object-oriented programming, routines are separate entities that include both data and functions, or methods, for processing the data. Object-oriented programming differs from traditional structured programming, which relied heavily on subroutines, and in which data and the routines that handled the data were usually strictly segregated. Structural programming is organized in terms of computer details, such as files, record, and sorting algorithms; object oriented programming is organized in terms of the physical entities, the "objects," that exist in the environment in which the problem is to be solved. The data and methods stored in a object represent, respectively, the attributes of the physical object and the actions a object is capable of car-

rying out. Objects are related to classes; an object is said to be an instance of a class. A class is not an object, but a template that instructs a compiler how to construct an actual object.

Those of ordinary skill in the art of modern programming are familiar with object-oriented concepts such as encapsulation, inheritance and polymorphism. Encapsulation refers to combining data and functions, or methods, that process the data into a single object, wherein the internal details of the methods and data may be largely hidden. Inheritance refers to a derived class inheriting the features and capabilities of the base, or parent, class. The derived class can then be modified to suit a particular need. Polymorphism allows objects whose properties differ to be able to function together. That is, the interfaces through which objects exchange messages are standardized and not dependent on the exact internal working of an object. Because of encapsulation, polymorphism and inheritance, object oriented-programming is a modular approach that reduces the amount of coding a designer of a new system or function needs to perform. Objects can be joined in a coherent procedure in which objects can be added, modified or deleted without a designer having to rewrite a whole application. A considerable amount of code written by others is reusable, and is often reused. For example, the Microsoft Foundation Class (MFC) Library is a set of more than 100 classes designed to help C++ programmers write Windows-based programs. The present invention built upon classes from the MFC version 4.1 library.

Accordingly, described herein are only those class definitions, and code segments, necessary for one of ordinary skill in the relevant programming arts, possessed of all the teachings herein, to make and use the invention. Classes, objects and codes well known to, or easily created by, those of ordinary skill in the art, or available in published libraries, such as MFC library, are not described, or if described, are only briefly discussed.

The present invention provides for a sophisticated property sheet system that supports modeless operation and tear-away property pages. The system uses the standard system software components illustrated in FIGURE 7. The components include the application frame window 46 and the property sheet system 48. In general, property sheet systems are implemented using three components: a property frame 48A, a property sheet 48B, and one or many property pages 48C. These three components 48A-48C are related. The property pages 48C are the children of the property sheet 48B; the property sheet 48B is the child of the property frame 48A; and the property system 48 is owned by the application's main frame window 46. Because the property frame 48A is typically a top-level window, it generally cannot be a child of another window, but it can be owned.

The present invention extends the standard property sheet system, such as that in FIGURE 7, by extending the standard classes to allow the tearing of property pages, the creation of new property sheet systems, and the docking of torn property pages into existing property sheet systems. The classes are illustrated in FIGURE 8.

The application frame window class 50, labeled CACRFrameWnd in FIGURE 8, provides management support for the property sheet system 55. The application frame window class 50 provides property page dock support, property page availability handling, and property sheet system creation. The application flame window 50 constructs the property sheet system 55 with a creation mask that indicates which property pages 56 to create. Additionally, the application frame window 50 provides the notification interface used by the property sheet system 55 when property or selection information has changed in a property page 36.

The property sheet system 55 is comprised of three distinct parts. The first and least significant of these components is the property sheet frame class 52. The property sheet frame 52 is a mini-frame window that is used to house the property sheet system 55 window. The property sheet frame 52 allows the property sheet system 55 window and pages 56 to be operated in an modeless fashion. The property sheet frame 52 creates the property sheet 54, hides itself when closed, and provides access methods to its owned property sheet 54.

The property sheet class 54 provides management support for the key property pages 56. This support includes property page creation, property page docking support, and property page update facilities.

The individual property pages 56 constitute the final component of the property sheet system 55. Each property page 56 provides specific edit functions for a portion of the data exchanged with an application program. Each property page is derived from a base property page that provides application-specific property page functions. The base property page provides tear-away page support, and facilities to update a keyboard or be updated from a keyboard. The base property page also provides an iterative mechanism for setting property information for multiple keys of keypad annunciator 16 in FIGURE 2 to facilitate disabling of controls.

Further elaboration is provided below, in a separate section on each of the components. Class definitions and a description of the instance variables and methods for the property frame class, property sheet class, and property page classes are provided. Important code sections for the property sheet and property page classes are listed at the end of the specification, and additional code, including code pertaining to modification of the application frame window class is provided in an appendix. Coding for the property sheet frame class is not provided.

Application Frame Window Class

Coding for portions of the application frame window class is provided in the appendix.

Property Sheet Frame Class - CPropertySheetFrame:CMiniFrameWnd

The property sheet frame 52 allows the property sheet system window and pages to be operated in a modeless fashion. The property sheet frame 52 can operate to create the property sheet, to hide itself when closed, and provides access methods to its owned property sheet.

As shown in FIGURE 8 and in FIGURE 9A, the CPropertySheetFrame 52 provides the harness window for the property sheet system. The CPropertySheetFrame 52 creates, destroys manages visibility, and facilities modeless operation of the property sheet system.

The CPropertySheetFrame class 52 is defined below in Tables 1 and 2.

Table 1 - CPropertySheetFrame Class Definition

Name	Parameters	Returns	Access	Virtual
CPropertySheetFrame	UINT wCreatePageMask	Nothing	public	no
~CPropertySheetFrame	void	Nothing	public	yes
GetPropertySheet	void	CACRPropertySheet*	public	no
OnCreate	LPCREATESTRUCT lpCreateStruct	int	protected	no
OnClose	void	void	protected	no
OnSetFocus	CWnd* pOldWnd	void	protected	no
OnActivate	UINT nState, CWnd* pWndOther, BOOL bMinimized	void	protected	no

Table 2 - CPropertySheetFrame Data Definition

Name	Type	Access
m_pACRPropertySheet	CACRPropertySheet*	private
m_wCreatePageMask	UINT	private

Below is a description of the data and methods of the CPropertySheetFrame 52.

Data:

m_pACRPropertySheet: This data member is a pointer to the property sheet system that is created and managed by this frame.

m_wCreatePageMask: This a data member s the initial page creation mask used to define this property sheet system. The page creation mask is a bit field that indicates to the property sheet system which property pages to create.

Methods:

CPropertySheetFrame(UINT wCreatePageMask): This method is the constructor for the CPropertySheetFrame class. The constructor will store the specified page creation mask internally and use it later during the creation process. The constructor will initialize the m_pACRPropertySheet member variable to NULL.

~CPropertySheetFrame(): This method is the destructor for the CPropertySheetFrame class. The destructor will destroy the m_pACRPropertySheet member variable.

GetPropertySheet(): This method returns the m_pACRPropertySheet pointer.

OnCreate(LPCREATESTRUCT lpCreateStruct): This method handles the creation of the frame window. This class customizes the creation process by creating the property sheet system at this time. The property sheet system is created with the page create mask, m_wCreatePageMask, that was specified during class creation.

OnClose(): This is a method that intercepts the normal window close behavior. The default behavior of the close process is to destroy the window and this class. This method simply hides the window. The destruction of this class is the responsibility of the CACRFrameWindow.

OnSetFocus(CWnd* pOldWnd): This is a method that forwards the focus handling to the embedded property sheet system if one is defined.

OnActivate(UINT nState, CWnd* pWndOther, BOOL bMinimized): This is a method that forwards the activation handling to the embedded property sheet system if one is defined.

Property Sheet Class - CACRPropertySheet:CPropertySheet

The CACRPropertySheet 54, illustrated in FIGURES 8 and 9B, provides support for the key property pages. CACRPropertySheet customizes the default property sheet behavior by providing automatic property page creation, property page docking support, and property page update facilities. The CACRPropertySheet class 54 is defined below in Tables 3 and 4.

Table 3 - CACRPropertySheet Class Definition

Name	Parameters	Returns	Access	Virtual
CACRPropertySheet	CWnd* pWndParent, UINT wPageCreateFlag	Nothing	public	no
~CACRPropertySheet	void	Nothing	public	yes
OnCreate	LPCREATSTRUCT lpCreateStruct	int	protected	no
CreatePage	UINT wPageMask	void	public	no
PostNcDestroy	void	void	public	no
DestroyPage	UINT wPageMask	void	public	no
IsPageEnabled	UINT wPageMask	BOOL	public	no
GetPageMask	void	UINT	public	no
FindPage	UINT wPageMask	CPropertyPage*	public	no
TearOffPage	UINT wPageMask, LPRECT pTearOffRect	void	public	no
UpdatePageControl	void	void	public	no

Table 4 - CACRPropertySheet Type Definitions

Mask	Value
NO_PAGES	0x00000000
PAGE_ACTIONS	0x00000001
PAGE_DISPLAYS	0x00000002
PAGE_PROGRAMS	0x00000004
PAGE_COMMANDS	0x00000008
PAGE_ALARM_LIST	0x00000010
PAGE_ALARM_COLOR	0x00000020
LEGAL_PAGES_ONLY	0x0000003f
ALL_AVAILABLE_PAGES	0xffffffff

Masks:

The following masks are used to define property pages that are used in the CACRProperySheet implementation.

NO_PAGES: Specifies no pages.
PAGE_ACTIONS: Specifies the Alarm Actions Property Page.
PAGES_DISPLAYS: Specifies the Display Property Page.
PAGE_PROGRAMS: Specifies the Program Property Page.

PAGE_COMMANDS: Specifies the Command Property Page.
PAGE_ALARM_LIST: Specifies the Alarm List Property Page..
PAGE_ALARM_COLOR: Specifies the Color Property Page.
LEGAL_PAGES_ONLY: Specifies all the legal Property Pages.
ALL_AVAILABLE_PAGES: Specifies all pages, legal or otherwise.

Methods:

CACRPropertySheet(CWnd* pWndParent, UINT wPageCreateFlag): This is the constructor for the property sheet system. The constructor will invoke the CreatePage member with the specified page create mask.

~CACRPropertySheet() This is the destructor for CACRPropertySheet. The destructor will delete all ed property pages.

OnCreate(LPCREATESTRUCT lpCreateStruct): The OnCreate handler automatically snaps this window to the client region of its parent, typically the CPropertySheetFrame.

PostNcDestroy(); This method calls the default PostNcDestroy handler and then deletes itself.

CreatePage(UINT wPageMask): This method will creates the property pages specified by the page mask. All created property pages are automatically added to this property sheet.

DestroyPage(UINT wPageMask): This method will find the page with the specified mask, remove it from the property sheet system, and destroy it. The page mask must specify a single page for this method.

IsPageEnabled(UINT wPageMask): This method will determine it the page specified by the page mask is present in this property sheet system. This method will return TRUE if the page is found, FALSE otherwise.

GetPageMask(void): This method will return the page mask for all the pages ed in this property sheet system.

FindPage(UINT wPageMask): This method will find the page specified by the page mask. This method will return the specified page if it is found. NULL otherwise.

TearOffPage(UINT wPageMask): This method will request the main frame window to create a property sheet sys-

tem in the specified page mask, remove the specified page from the property sheet system, and if the property page removed is the last on kill this class.

UpdatePageControl(void): This method provides a work around to a MFC 3.0 bug concerning the automatic redraw of the page control used in the property sheet system.

Property Page - CACRPropertyPage: CPropertyPage

The CACRPropertyPage 56, as illustrated in FIGURES 8 and 9C, provides the core behavior for the key property sheet system for the process control application program. The key property sheet system is used to configure the keys, of the keypad annunciator 16 in FIGURE 2. CACRPropertyPage 56 provides tear-away page support and facilitates setting of properties from and to a target data source.

The CACRPropertyPage class is defined below in Tables 5 and 6.

Table 5 - CACRPropertyPage Class Definition

Name	Parameters	Returns	Access	Virtual
CACRPropertyPage	UINT wPageID, UINT nIDTemplate, UINT nIDCaption	Nothing	public	no
CACRPropertyPage	LPCSTR lpszTemplateName, UINT nIDCaption	Nothing	public	no
~CACRPropertyPage	void	Nothing	public	yes
GetPageID	void	UINT	public	no
BeginPropertySet	void	void	public	yes
EndPropertySet	void	void	public	yes
GetPropertySheet	void	CACRPropertySheet	public	no
ClipRectToWindow	HWND hWnd, LPRECT pRect	void	protected	no
Track	HDC hDC, LPRECT pTrackRect, BOOL bKillTrack	void	protected	no
TearOffDialog	LPRECT pTearOffRect	void	protected	no
OnLButtonDown	UINT nFlags, CPoint point	void	protected	no
OnLButtonUp	UINT nFlags, CPoint point	void	protected	no
OnMouseMove	UINT nFlags, CPoint point	void	protected	no

Table 6 - CACRPropertyPage Data Definition

Name	Type	Access
m_rTrackRect	RECT	protected
m_bIsTracking	BOOL	protected
m_bIsTearable	BOOL	protected
m_bDitherLast	BOOL	protected
m_sizeLast	SIZE	protected
m_rectLast	RECT	protected
m_ptStartPoint	POINT	protected
m_wPageID	UINT	protected

Data:

m_rTrackRect: This variable s the property page tracking rectangle during a page tear operation.

m_bIsTracking: This variable indicates that the property page is in the process of being torn off.

m_bIsTearable: This variable indicates that the current property page is tearable.

m_bDitherLast: This variable indicates the state of the last drag track render operation. To maximize visibility the track rectangle is rendered in a combination of dithered states and non-dithered states.

m_sizeLast: This variable s the size of the last track render operation.

m_rectLast: This variable s the rectangle that was previously rendered during the drag operation.

m_ptStartPoint: This variable s the starting point of the drag operation. The start point will maintain the relative position of the drag rectangle to the mouse.

m_wPageID: This variable s the page id of this property page. The page id is defined on creation of the property page.

Methods:

CACRPropertyPage(UINT wPageID,UINT nIDTemplate,UINT nIDCaption): This is the constructor for the CACRPropertyPage dialog. The constructor requires specification of the page ID, a dialog template, and a caption.

CACRPropertyPage(UINT wPageID ,LPCSTR lpszTemplateName,UINT nIDCaption): This is a constructor for the CACRPropertyPage dialog. This version of the constructor allows for the specification of the template using a string.

~ CACRPropertyPage(): This is the destructor for the CACRPropertyPage class.

GetPageID(void): This method returns the m_wPageID member variable.

BeginPropertySet(void): This method provides a hook for derived property page classes to initialize data state information to properly reflect data values under multiple keys.

EndPropertySet(void): This method provides a hook for derived property pages to end the property setting process by reflecting collected state information into the property page controls.

GetPropertySheet(void): This method returns the parent property sheet. This method will ASSERT if the parent property sheet is not of CACRPropertySheet style.

ClipRectToWindow(HWND hWnd,LPRECT pRect): This method will clip the specified rectangle to the specified window. The method will maintain the size of the specified rectangle if possible.

Track(HDC hDC,LPRECT pTrackRect,BOOL bKillTrack=FALSE): This method performs tracking for the property page drag/tear operation.

TearOffDialog(LPRECT pTearOffRect): This method performs the property page tear operation. It determines it the specified rectangle represents a tearable state and if so performs the tear operation.

OnLButtonDown(UINT nFlags, CPoint point): This method handles the left mouse button down processing. This method will start the property page tear operation.

OnLButtonUp(UINT nFlags, CPoint point): This method handles the left mouse button up processing. This method will end the property page tear operation.

OnMouseMove(UINT nFlags, CPoint point): This method handles mouse move processing. This method will drag

the page tear rectangle if the mouse button is pressed and the page tear operation is underway.

Key Property Pages

In the application described here, wherein the property pages are used to exchange information about, and to configure, the keys of keypad annunciator 16 in FIGURE 2, the key property pages are derived from the CACRPropertyPage base class, as shown in FIGURE 10. The property pages provide specific edit operations for the key data.

Each property page will be implemented in a similar fashion. In general, each property page will provide the following interface, as set forth in Table 7.

Table 7 - CACRGenericEditPage Class Definition

Name	Parameters	Returns	Access	Virtual
CACRGenericEditPage	UINT wPageID, UINT nIDTemplate, UINT nIDCaption	Nothing	public	no
CACRGenericEditPage	LPCSTR lpszTemplateName, UINT nIDCaption	Nothing	public	no
~CACRGenericEditPage	void	Nothing	public	yes
BeginPropertySet	void	void	public	yes
EndPropertySet	void	void	public	yes

Code sections for the property sheet and property page classes, as well as code sections pertaining to modification of the application frame window class, are provided in an appendix. The code is written in the C++ programming language, but can be used as the operational foundation for other implementations.

APPENDIX

C++ Code for CACRPropertySheet Class

```
//**************************************************************************
//**************************************************************************
//
// Class:       CreatePage
// Description: Create the property page for this mask and add it to the system.
//              This routine will create multiple pages if multiple masks are
//              specified.
//
// Returns:     Bool
//
//**************************************************************************

    1 CModelessPropertySheet::CreatePage(UINT wPageMask)

     CPropertyPage*  pPage;

                                        //----------------------------------------
                                        // Determine if the individual page types
                                        // are defined and create and add them to the
                                        // sheet.
                                        //----------------------------------------
    if (wPageMask & PAGE_ACTIONS)
    {
        pPage = new CAlmActions();
        AddPage(pPage);
    }
    if (wPageMask & PAGE_DISPLAYS)
    {
        pPage = new CDisplayListPage();
        AddPage(pPage);
```

```
    }
    if (wPageMask & PAGE_PROGRAMS)
    {
        pPage = new CProgramListPage();
        AddPage(pPage);
    }
    if (wPageMask & PAGE_COMMANDS)
    {
        pPage = new CCommandDlg();
        AddPage(pPage);
    }
    if (wPageMask & PAGE_ALARM_LIST)
    {
        pPage = new CAlarmList();
        AddPage(pPage);
    }
    if (wPageMask & PAGE_ALARM_COLOR)
    {
        pPage = new CNewAlarmColor();
        AddPage(pPage);
    }
}


//**************************************************************************
//**************************************************************************
//
//  Method:       DestroyPage
//  Description: This method toasts the property page associated with this mask.
//               This is used when a property page has been removed from one sheet
//               and put into another sheet.
//
//  Returns:     Bool
//
//**************************************************************************

void CModelessPropertySheet::DestroyPage(UINT wPageMask)
{
    CPropertyPage*  pPropertyPage;
                                    //-----------------------------------------
                                    // Get It. Remove It. KILL It.
                                    //-----------------------------------------
    if ((pPropertyPage = FindPage(wPageMask)) != NULL)
    {
        RemovePage(pPropertyPage);
        delete pPropertyPage;
    }
}


//**************************************************************************
//**************************************************************************
//
// Class:        FindPage
// Description: Get the property page pointer based on a mask
//
// Returns:     Bool
//
//**************************************************************************

opertyPage* CModelessPropertySheet::FindPage(UINT wPageMask)

    CACRPropPage*   pPage = NULL;

    for (int i=0; i<GetPageCount(); i++)
    {
                                    // Am I the guy we want?


//**************************************************************************
//**************************************************************************
//
// Class:        CModelessPropertySheet
// Description: This defines the core modeless property sheet.
//
//**************************************************************************
```

```
  #ifine     NO_PAGES              0x00000000
#define      PAGE_ACTIONS          0x00000001
#define      PAGE_DISPLAYS         0x00000002
#define      PAGE_PROGRAMS         0x00000004
#define      PAGE_COMMANDS         0x00000008
#define      PAGE_ALARM_LIST       0x00000010
#define      PAGE_ALARM_COLOR      0x00000020

#ifdef SOFTPANEL
#define      LEGAL_PAGES_ONLY      0x0000003f
#define      ALL_AVAILABLE_PAGES   0xffffffff
#else
#define      LEGAL_PAGES_ONLY      0x0000001f
#define      ALL_AVAILABLE_PAGES   0xffffffdf
#endif

class CModelessPropertySheet;
class CKeyboardCore;
class CACRKey;


class CModelessPropertySheet : public CPropertySheet
{
private:
    CKeyboardCore*  m_pOwnerKeyboard;

public:
    DECLARE_DYNAMIC(CModelessPropertySheet)
                    CModelessPropertySheet      (CWnd* pWndParent,UINT wPageCreateFlag);
                    ~CModelessPropertySheet     ();

// Attributes
    void            CreatePage                  (UINT wPageMask);
    void            DestroyPage                 (UINT wPageMask);
    BOOL            IsPageEnabled               (UINT wPageMask);
    UINT            GetPageMask                 (void);
    CPropertyPage*  FindPage                    (UINT wPageMask);
    void            TearOffPage                 (UINT wPageMask,LPRECT pTearOffRect);
    void            UpdatePageControl           (void);

    void            SetPropertiesFromKeyboard   (CKeyboardCore* pKeyboard);


     if ( ((pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
          (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage))) &&
          (pPage->GetPageID() & wPageMask) )
     {
         return pPage;
     }
  }
  ASSERT(pPage != NULL);              // If we are asking for a NULL page some
                                      // bogus is going on.
  return pPage;
}


//****************************************************************************
//****************************************************************************
//
// Class:       IsPageEnabled
// Description: This routine loops through the pages associated with this
//              dialog system and determines if they are present.
//
// Returns:     Bool
//
//****************************************************************************

BOOL CModelessPropertySheet::IsPageEnabled(UINT wPageMask)
{
    CACRPropPage*   pPage = NULL;

    for (int i=0; i<GetPageCount(); i++)
    {
                                        // Am I the guy we want?
        if ( ((pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
             (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage))) &&
             (pPage->GetPageID() & wPageMask) )
        {
            return TRUE;
        }
    }
    return FALSE;
```

```
//**************************************************************************
//**************************************************************************
//
// Class:       GetPageMask
// Description: This routine loops through the pages associated with this
//              dialog system and collects their page masks.
//
// Returns:     UINT
//
//**************************************************************************

UINT CModelessPropertySheet::GetPageMask(void)
{
    UINT            wPageMask = 0;
    CACRPropPage*   pPage = NULL;

    for (int i=0; i<GetPageCount(); i++)
    {
                                        // Am I the guy we want?
        if ( ((pPage = (CACRPropPage*)GetPage(i)) != NULL) &&
             (pPage->IsKindOf(RUNTIME_CLASS(CACRPropPage))) )
            wPageMask |= pPage->GetPageID();
    }
    return wPageMask;
}


//**************************************************************************
//**************************************************************************

   Class:       PageChanged
// Description: This routine is called by the Property pages when something
//              has occured that changes their data.  We will tell the keyboard
//              core so he may update his information.
//
// Returns:     Nothing



//**************************************************************************
//**************************************************************************
//
// Method:      TearOffPage
// Description: This routine removes the requested page from this sheet
//              and tells the frame to create a new one.
//
// Returns:     Nothing
//
//**************************************************************************

   d CModelessPropertySheet::TearOffPage(UINT wPageMask,LPRECT pTearOffRect)

    CFrameWnd*  pNewFrame;

   if ((pNewFrame = (CFrameWnd*)ACRGetMainWindow()->AddPropertySheet(GetOwnerKeyboard(),wPageMask,pTearOf
t)) != NULL)
    {

                                    //-------------------------------------------
                                    // If it is the last page we should commit
                                    // sucide
                                    //-------------------------------------------
        if (GetPageCount() == 1)
            GetParent()->DestroyWindow();
                                    //-------------------------------------------
                                    // Get It. Remove It. KILL It.
                                    //-------------------------------------------
        else
        {
            DestroyPage(wPageMask);

                                    //-------------------------------------------
                                    // Put the new one on top
                                    //-------------------------------------------
            if (pNewFrame != (CFrameWnd*)GetParent())
                pNewFrame->BringWindowToTop();
        }
    }
  }
```

```
//*****************************************************************************
//
// Method:      UpdatePageControl
// Description: This method forces the control used in the property pages to
//              be redrawn.
//
// Returns:     Nothing
//
//*****************************************************************************

void CModelessPropertySheet::UpdatePageControl(void)
{
    m_tabRow.SetCurSel(m_tabRow.GetItemCount()-1);
}
```

C++ Code for CACRPropertyPage Class

```
//*****************************************************************************
//*****************************************************************************
//
// Class:       CACRPropPage
// Description: This defines the alarm color dialog.
//
//*****************************************************************************

class   CKeyboardCore;
class   CACRKey;
class   CModelessPropertySheet;
class   CACRDocument;

class CACRPropPage : public CPropertyPage

    DECLARE_DYNAMIC(CACRPropPage)

protected:
    RECT                m_trackRect;
    BOOL                m_bIsTracking;
    BOOL                m_bIsTearable;
    BOOL                m_bDitherLast;
    SIZE                m_sizeLast;
    RECT                m_rectLast;
    POINT               m_ptStartPoint;
    UINT                m_wPageID;              // System defined user mask
    BOOL                m_bEnableTracking;
    BOOL                m_SelectedKeyCount;
    BOOL                m_bIsInited;

// Construction
public:
                        CACRPropPage            (UINT wPageID,UINT nIDTemplate,UINT nIDCaption=0);
                        CACRPropPage            (LPCSTR lpszTemplateName,UINT nIDCaption);
                        ~CACRPropPage           ();

    virtual BOOL        OnSetActive             ();
    virtual BOOL        CanClose                (void){return TRUE;};
    UINT                GetPageID               (void);
    virtual void        SetPropertiesFromKeyboard   (CKeyboardCore* pKeyboard);
    virtual void        SetKeyboardFromProperties   (CKeyboardCore* pKeyboard,LONG lHint=0L);

    virtual void        BeginPropertySet        (void){};
    virtual void        SetPropertyFromKey      (CACRKey* pKey){}
    virtual void        SetKeyFromProperty      (CACRKey* pKey,LONG lHint=0){}
    virtual void        EndPropertySet          (void){}
    virtual UINT        GetUndoOperationID      (LONG lHint){return STR_ID_EDIT_UNDO;};

    CModelessPropertySheet* GetPropertySheet    (void);
    CACRDocument*       GetDocument             (void);
    CKeyboardCore*      GetKeyboard             (void);

// Dialog Data
    //{{AFX_DATA(CACRPropPage)
```

```
        //}}AFX_DATA

    // Overrides
        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CACRPropPage)
        protected:
        //}}AFX_VIRTUAL


    // Implementation
    protected:
        void                ClipRectToWindow        (HWND hWnd,LPRECT pRect);
        void                Track                   (HDC hDC,LPRECT pTrackRect,BOOL bKillTrack=FALSE);
        void                TearOffDialog           (LPRECT pTearOffRect) ;

        // Generated message map functions
        //{{AFX_MSG(CACRPropPage)
        virtual BOOL        OnInitDialog            ();
        afx_msg void        OnLButtonDown           (UINT nFlags, CPoint point);
        afx_msg void        OnLButtonUp             (UINT nFlags, CPoint point);
        afx_msg void        OnMouseMove             (UINT nFlags, CPoint point);
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
    };

    inline UINT CACRPropPage::GetPageID(void)
    {
        return m_wPageID;
    }
    #endif


//*****************************************************************************
//*****************************************************************************
//
// Method:      CACRPropPage::CACRPropPage
// Description: This routine constructs the ACFG Property Page
//
//   Returns:     Nothing
//
//*****************************************************************************

CACRPropPage::CACRPropPage(UINT wPageID,UINT nIDTemplate,UINT nIDCaption) : CPropertyPage(nIDTemplate,nIDC
on)
{
    m_wPageID           = wPageID;
    m_bEnableTracking   = FALSE;
    m_bIsTracking       = FALSE;
    m_bIsTearable       = FALSE;
    m_bDitherLast       = FALSE;
    m_sizeLast.cx       = m_sizeLast.cy = 0;
    ::SetRectEmpty(&m_rectLast);
    m_SelectedKeyCount  = 0;
    m_bIsInited         = FALSE;

    //{{AFX_DATA_INIT(CACRPropPage)
        // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}


//*****************************************************************************
//*****************************************************************************
//
// Method:      CACRPropPage::CACRPropPage
// Description: This routine constructs the ACFG Property Page
//
// Returns:     Nothing
//
//*****************************************************************************

CRPropPage::CACRPropPage(LPCSTR lpszTemplateName,UINT nIDCaption) : CPropertyPage(lpszTemplateName,nIDC

{
    //{{AFX_DATA_INIT(CACRPropPage)
        // NOTE: the ClassWizard will add member initialization here
    //}}AFX_DATA_INIT
}
```

```
//**************************************************************************
//**************************************************************************
//
// Class:       OnLButtonDown
// Description: Tear away this popup and make a modeless dialog out of it .
//
// Returns:     BOOL
//
//**************************************************************************

void CACRPropPage::OnLButtonDown(UINT nFlags, CPoint point)
{
    HWND    hDesktop = ::GetDesktopWindow();

    if (GetParent()->IsWindowVisible() && hDesktop)
    {
        HDC     hDC = ::GetDCEx(hDesktop,NULL,DCX_PARENTCLIP);

                                    // Render the popup outline
        GetClientRect(&m_rTrackRect);
        ::MapWindowPoints(GetSafeHwnd(),hDesktop,(LPPOINT)&m_rTrackRect,2);
        m_ptStartPoint = point;
        m_bIsTracking  = TRUE;
        m_bIsTearable  = FALSE;
        m_bDitherLast  = FALSE;
        ::SetRectEmpty(&m_rectLast);

        m_sizeLast.cx  = m_sizeLast.cy = 0;

        ::SetCapture(GetSafeHwnd());

                                    // Render the popup outline
        Track(hDC,&m_rTrackRect);

                                    // Release the display contex
        ::ReleaseDC(hDesktop,hDC);
    }
    else
        CPropertyPage::OnLButtonDown(nFlags, point);
}


//**************************************************************************
//**************************************************************************
//
// Class:       OnLButtonUp
// Description: If tracking a popup make a modeless dialog out of it.
//
// Returns:     BOOL
//
//**************************************************************************

void CACRPropPage::OnLButtonUp(UINT nFlags, CPoint point)
{

    if (m_bIsTracking)
    {
        RECT    rClientRect;
        GetClientRect(&rClientRect);
        HWND    hDesktop = ::GetDesktopWindow();
        HDC     hDC;

        ASSERT(hDesktop);

        hDC = ::GetDCEx(hDesktop,NULL,DCX_PARENTCLIP);

                                    // Undraw and release dc
        Track(hDC,&m_rTrackRect,TRUE);

        ::ReleaseDC(hDesktop,hDC);


                                    // Create the modeless property
                                    // sheet dialog here.
        if ((m_bIsTearable) && !::PtInRect(&rClientRect,m_ptStartPoint))
            TearOffDialog(&m_rTrackRect);
```

```
        m_bIsTracking = FALSE;
        m_bIsTearable = FALSE;
        ::ReleaseCapture();
    }
    else
        CPropertyPage::OnLButtonUp(nFlags, point);
}


//**************************************************************************
//**************************************************************************
//
// Class:       OnMouseMove
// Description: Track the popup if it being torn away.
//
// Returns:     BOOL
//
//**************************************************************************

void CACRPropPage::OnMouseMove(UINT nFlags, CPoint point)
{
    RECT    rClientRect;

    GetClientRect(&rClientRect);
    if (m_bIsTracking)
    {
        HWND    hDesktop = ::GetDesktopWindow();
        HDC     hDC;

        ASSERT(hDesktop);
        hDC = ::GetDCEx(hDesktop,NULL,DCX_PARENTCLIP);

        if (!::PtInRect(&rClientRect,point))
        {
            int     nDX,nDY;

//          if (m_bIsTearable)
//              Track(hDC,&m_rTrackRect);   // Undraw
                                            // Recalculate the tracking rectangle
            nDX = point.x - m_ptStartPoint.x;
            nDY = point.y - m_ptStartPoint.y;

            ::OffsetRect(&m_rTrackRect,nDX,nDY);
            ClipRectToWindow(hDesktop,&m_rTrackRect);

            m_ptStartPoint = point;     // Remeber this point

            Track(hDC,&m_rTrackRect);   // Redraw
            ::ReleaseDC(hDesktop,hDC);  // Release

            m_bIsTearable = TRUE;
        }
        else if (m_bIsTearable)
        {
            Track(hDC,&m_rTrackRect,TRUE);  // Undraw
            m_bIsTearable = FALSE;
        }
    }
    else
        CPropertyPage::OnMouseMove(nFlags, point);
}


#define CX_BORDER   1
#define CY_BORDER   1
#define TITLEHEIGHT 16

//**************************************************************************
//
// Class:       Track
// Description: Render the tracking rectangle in XNOR mode on the specified
//              display context.
//
// Returns:     BOOL
//
//**************************************************************************
```

```
void CACRPropPage::Track(HDC hDC,LPRECT pTrackRect,BOOL bKillTrack)
{
    CDC*    pDC=CDC::FromHandle(hDC);
                        // default to thin frame
    CSize   size(CX_BORDER, CY_BORDER);
                        // determine new rect and size
    CRect   rect;
    CBrush* pWhiteBrush  = CBrush::FromHandle((HBRUSH)::GetStockObject(WHITE_BRUSH));
    CBrush* pDitherBrush = CDC::GetHalftoneBrush();
    CBrush* pBrush = pWhiteBrush;
    HCURSOR hCursor;

                        // Set track rect size
    rect = *pTrackRect;

                        // Use thick frame if we are not over a docking
                        // site

    if (ACRGetMainWindow()->CanDock(&rect) == NULL)
    {
        size.cx = GetSystemMetrics(SM_CXFRAME) - CX_BORDER;
        size.cy = GetSystemMetrics(SM_CYFRAME) - CY_BORDER;

        pBrush = pDitherBrush;
        hCursor = AfxGetApp()->LoadCursor(IDC_NOTDOCKED);
    }
    else
        hCursor = AfxGetApp()->LoadCursor(IDC_DOCKED);

    if (bKillTrack || !m_bIsTearable)
    {
        size.cx = size.cy = 0;
        hCursor = AfxGetApp()->LoadStandardCursor(IDC_ARROW);
    }
    ::SetCursor(hCursor);

                        // draw it and remember last size
    pDC->DrawDragRect(&rect, size, &m_rectLast, m_sizeLast,
                      pBrush, m_bDitherLast ? pDitherBrush : pWhiteBrush);
    m_rectLast = rect;
    m_sizeLast = size;
    m_bDitherLast = (pBrush == pDitherBrush);
}


//**************************************************************************
//**************************************************************************
//
// Class:       ClipRectToWindow
// Description: This routine clips the specified rectangle to the specified
//              window.
//
// Returns:     BOOL
//
//**************************************************************************

void CACRPropPage::ClipRectToWindow(HWND hWnd,LPRECT pRect)
{
    RECT    rClientRect;
    int     nWidth,nHeight;

#ifndef WU_APP
    ::GetClientRect(hWnd,&rClientRect);
  .se
    ::GetWindowRect(hWnd,&rClientRect);
#endif
    nWidth = pRect->right - pRect->left;
    nHeight = pRect->bottom - pRect->top;

                                    // Clip top and bottom


     if (pRect->top < rClientRect.top)
     {
         pRect->top    = rClientRect.top;
         pRect->bottom = pRect->top + nHeight;
     }
     else if (pRect->bottom > rClientRect.bottom)
     {
         pRect->bottom = rClientRect.bottom;
         pRect->top    = pRect->bottom - nHeight;
     }
```

```
                                        // Clip left and right
    if (pRect->left < rClientRect.left)
    {
        pRect->left   = rClientRect.left;
        pRect->right  = pRect->left + nWidth;
    }
    else if (pRect->right > rClientRect.right)
    {
        pRect->right = rClientRect.right;
        pRect->left  = pRect->right - nWidth;
    }
}

//*************************************************************************
//*************************************************************************
//
// Class:       TearOffDialog
// Description: This routine creates another version of this dialog, only in
//              modeless form.
//
// Returns:     BOOL
//
//*************************************************************************

#define MIN_NEW_DELTAX  5
#define MIN_NEW_DELTAY  5

   i CACRPropPage::TearOffDialog(LPRECT pTearOffRect)
{
    HWND                    hDesktop = ::GetDesktopWindow();
    RECT                    rClientRect;
    CModelessPropertySheet* pPropertySheet = GetPropertySheet();

                                        //---------------------------------------
                                        // Convert the client coords to the
                                        // desktop coords to determine if the
                                        // tracking window has moved enough to
                                        // tear it off.
                                        //---------------------------------------
    GetClientRect(&rClientRect);
    ::MapWindowPoints(GetSafeHwnd(),hDesktop,(LPPOINT)&rClientRect,2);

                                        //---------------------------------------
                                        // Get this pages ID, create a new property
                                        // sheet with it, and destroy the page.
                                        //---------------------------------------
    if (pPropertySheet)
        pPropertySheet->TearOffPage(GetPageID(),pTearOffRect);
}


//*************************************************************************
//*************************************************************************
//
// Class:       GetPropertySheet
// Description: This routine returns the parent property sheet in a digestable
//              format.
//
// Returns:     BOOL
//
//*************************************************************************

CModelessPropertySheet* CACRPropPage::GetPropertySheet(void)
{
    CWnd*                   pWnd = GetParent();
    CModelessPropertySheet* pPropertySheet;

    pPropertySheet = (CModelessPropertySheet*)pWnd;
    if (pPropertySheet && (pPropertySheet->IsKindOf(RUNTIME_CLASS(CModelessPropertySheet))) )
        return pPropertySheet;

    return NULL;
```

C++ Code for Application Frame

```
//*****************************************************************************
//*****************************************************************************
//
// Method:      AddPropertySheet
// Description: This routine creates a frame window with property sheet, and
//              populates it with available property pages.
//
// Returns:     CPropertySheetFrame*
//
//*****************************************************************************

CPropertySheetFrame* CACRMainFrame::AddPropertySheet(CKeyboardCore* pActiveKeyboard,UINT wPageMask,LPRECT pPro
pertyRect)
{
    CPropertySheetFrame*    pPropertySheetFrame=NULL;

                                    // A valid keyboard is required
    ASSERT(pActiveKeyboard != NULL);

                                    // If the user is creating a standard property frame
                                    // we should give them all the available pages
    if (wPageMask == ALL_AVAILABLE_PAGES)
        wPageMask = GetAvailablePageMask();

    if (wPageMask == NO_PAGES)
        return FALSE;

                                    // Determine if we can dock this sheet to an existing
                                    // frame.  We are in docking mode if the pPropertyRect is
                                    // specified.
    if (pPropertyRect && ((pPropertySheetFrame=CanDock(pPropertyRect)) != NULL))
    {
        CModelessPropertySheet*     pPropertySheet = pPropertySheetFrame->GetPropertySheet();
        pPropertySheet->CreatePage(wPageMask);
#ifndef DONTREINIT
        pPropertySheet->SetOwnerKeyboard(pActiveKeyboard);
        pPropertySheet->SetPropertiesFromKeyboard(pActiveKeyboard);
#endif
        pPropertySheet->UpdatePageControl();
    }
    else
    {
                                        // Watch out for those exceptoins
        TRY
        {
            CRect                   rect(0,0,0,0);
            CString                 strTitle;

                                        // Create this guy and position it.
            pPropertySheetFrame = new CPropertySheetFrame(wPageMask);


            strTitle.Empty();
            if (!pPropertySheetFrame->Create(NULL, strTitle,WS_POPUP | WS_CAPTION | WS_SYSMENU, rect, this))
                delete pPropertySheetFrame;

            UpdateKeyPropertyTitle(pPropertySheetFrame,pActiveKeyboard);

                                        // Initialize the data
            pPropertySheetFrame->GetPropertySheet()->SetOwnerKeyboard(pActiveKeyboard);
            pPropertySheetFrame->GetPropertySheet()->SetPropertiesFromKeyboard(pActiveKeyboard);


                                        // If a property rect is specified use it to
                                        // position this guy
            if (pPropertyRect)
                pPropertySheetFrame->SetWindowPos(NULL,pPropertyRect->left,pPropertyRect->top,0,0,SWP_NOSIZE|S
WP_NOZORDER);
            else
                pPropertySheetFrame->CenterWindow(ACRGetDialogCenterWindow());

            if (!pPropertySheetFrame->IsWindowVisible())
                pPropertySheetFrame->ShowWindow(SW_SHOW);
        }
```

```
        CATCH_ALL(e)
        {
            delete pPropertySheetFrame;
            pPropertySheetFrame = NULL;
        }
        END_CATCH_ALL
    }
    return pPropertySheetFrame;
}


//*****************************************************************************
//*****************************************************************************
//
// Method:      GetAvailablePageMask
// Description: This routine calculates the available page masks by enumerating
//              all the child property frames.

// Returns:     UINT
//
//*****************************************************************************

UINT CACRMainFrame::GetAvailablePageMask(void)
{
    UINT    wPageMask = LEGAL_PAGES_ONLY;
    CWnd*   pWnd = this;

                        // Loop through all the child window and find the
                        // frames and add their Page Mask to this page mask.
    while (pWnd)
    {
        if ( (pWnd->IsKindOf(RUNTIME_CLASS(CPropertySheetFrame))) &&
             (pWnd->GetParent()->GetSafeHwnd() == GetSafeHwnd()) )
        {
            CPropertySheetFrame*    pPropertyFrame = (CPropertySheetFrame*)pWnd;
            CModelessPropertySheet* pPropSheet = pPropertyFrame->GetPropertySheet();

            wPageMask &= ~pPropSheet->GetPageMask();
        }
        pWnd = pWnd->GetNextWindow(GW_HWNDPREV);
    }
    return wPageMask;
}


//*****************************************************************************
//*****************************************************************************
//
// Method:      CanDock
// Description: This routine determines if we dock the specified frame to
//              an existing frame.

// Returns:     CPropertySheetFrame* -> Docking frame if one if found.
//
//*****************************************************************************

#define     DOCKX_FUZZ  10

#define     DOCKY_FUZZ  10

CPropertySheetFrame* CACRMainFrame::CanDock(LPRECT pPropertyRect)
{
    CPropertySheetFrame*    pPropertySheetFrame=NULL;
    UINT                    wPageMask = LEGAL_PAGES_ONLY;
    CWnd*                   pWnd = this;
    RECT                    rTestRect = *pPropertyRect;


                        // reduce the test rect
    ::InflateRect(&rTestRect,-DOCKX_FUZZ,-DOCKY_FUZZ);

                        // Loop through all the child window and find the
                        // frames and add their Page Mask to this page mask.
    while (pWnd)
    {
        if ( (pWnd->IsKindOf(RUNTIME_CLASS(CPropertySheetFrame))) &&
             (pWnd->GetParent()->GetSafeHwnd() == GetSafeHwnd()) )
        {
            RECT    rWindowRect;
```

```
                    // Get the window rect for this frame;
            pWnd->GetWindowRect(&rWindowRect);

            if ( (rWindowRect.top < rTestRect.top) && (rWindowRect.bottom > rTestRect.bottom) &&
                 (rWindowRect.left < rTestRect.left) && (rWindowRect.right > rTestRect.right) )
                return (CPropertySheetFrame*)pWnd;
        }
        pWnd = pWnd->GetNextWindow(GW_HWNDPREV);
    }
    return NULL;
}


//*****************************************************************************
//*****************************************************************************
//
// Method:      SyncPropertySheets
// Description: This routine syncs the property sheets up with the current
//              selection state of the keyboard.
//
// Returns:     Nothing
//
//*****************************************************************************

void CACRMainFrame::SyncPropertySheets(CKeyboardCore* pKeyboard)
{
                            // Get the active keyboard if one is not specified.
    if (!pKeyboard)
        pKeyboard = GetActiveKeyboard();

                            // Sync each property frame if a keyboard is
                            // specified.
    if (pKeyboard)
    {
        FRAMEPOSITIONINFO   FramePositionInfo;

        FramePositionInfo.hWindowOnTopOf      = HWND_TOP;
        FramePositionInfo.hDeferPositionInfo  = ::BeginDeferWindowPos(10);
        FramePositionInfo.hParent             = GetSafeHwnd();
        FramePositionInfo.pKeyboard           = pKeyboard;
        FramePositionInfo.bInvalidate         = FALSE;
        FramePositionInfo.pScreenRect         = NULL;
        FramePositionInfo.wPropMode           = kPropSync;

        ::EnumWindows(ACREnumChildProc, (LPARAM)&FramePositionInfo);

        if (FramePositionInfo.hDeferPositionInfo)
            ::EndDeferWindowPos(FramePositionInfo.hDeferPositionInfo);
    }
}


//*****************************************************************************
//*****************************************************************************
//
// Method:      ClosePropertySheets
// Description: This method enumerates each property sheet and closes them.
//
// Returns:     Nothing
//
//*****************************************************************************

void CACRMainFrame::ClosePropertySheets(void)
{
                            // Sync each property frame if a keyboard is
                            // specified.
    FRAMEPOSITIONINFO   FramePositionInfo;

    memset(&FramePositionInfo,0,sizeof(FramePositionInfo));

    FramePositionInfo.hWindowOnTopOf      = HWND_TOP;
    FramePositionInfo.hDeferPositionInfo  = NULL;
    FramePositionInfo.hParent             = GetSafeHwnd();
    FramePositionInfo.pKeyboard           = NULL;
    FramePositionInfo.bInvalidate         = FALSE;
    FramePositionInfo.pScreenRect         = NULL;
    FramePositionInfo.wPropMode           = kPropClose;

    ::EnumWindows(ACREnumChildProc, (LPARAM)&FramePositionInfo);
}
```

Claims

1. Apparatus for computerised data processing, comprising

processor means for executing a program,
a memory element for storing the program,
a display monitor for providing a display,
selection means for selecting a property page of a first property sheet system, the first property sheet system including at least one property page displayable at a first position on the display monitor, and
creation means for creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

2. Apparatus for configuring a peripheral via a computer or configuring the intercommunication between a computer and a peripheral, the computer comprising;

processor means for executing a program,
a memory element for storing at least the program,
a display monitor for providing a display,
the processor means being adapted to display via the display monitor property sheet systems for receiving configuration settings data from a user, the processor means being further adapted to store said data in the memory element at least temporarily and to employ said data when communicating with the peripheral;
selection means for selecting a property page of a first property sheet system, the first property sheet system including at least one property page displayable at a first position on the display monitor, and
creation means for creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

3. The apparatus of Claim 1 or Claim 2 further including

removal means for removing said property page selected by said selection means from said first property sheet system such that said selected property page is not displayable by said first property sheet system.

4. The apparatus of Claim 3 further including

destruction means for destroying said first property sheet system when said first property sheet system does not include any property pages.

5. The apparatus of any preceding Claim further comprising

positioning means for varying and indicating on the display monitor said second display position at which said property page selected by said selection means and included in said second property sheet system is displayable by said second property sheet system.

6. The apparatus of any preceding claim further including

docking means for placing said property page selected by said selection means into a selected property sheet system for inclusion therein, said property page being displayable on the display monitor by said selected property sheet system.

7. The apparatus of Claim 6 further comprising

positioning means for varying and indicating the position of the display monitor of a symbol representative of a property page selected by said selection means, and
proximity means for determining a parameter responsive to the distance on the display monitor between said symbol and the position on the display monitor of a displayed property page of a property sheet system, and
wherein said parameter is less than a first selected value, said docking means in response to a first user input places said selected property page in said property sheet system displaying said displayed property page for inclusion therein and for display on the display monitor thereby.

8. The apparatus of Claim 7 wherein when said parameter is greater than a second selected value, said creation

means in response to a second user input displays said selected property page included in said second property sheet system at a position on the display monitor determined by the display position of said symbol.

9. The apparatus of Claim 7 further comprising

icon means for displaying on the display monitor an icon having at least a first display state, said icon displayed in said first display state when said parameter is less than said first selected value, said first display state for indicating to a user that said docking means, in response to said first user input, will place said selected property page into said property sheet system displaying said displayed property page.

10. The apparatus of Claim 9 wherein said icon means further comprises means for displaying said icon in a second display state when said parameter is greater than a second selected amount, said second display state for indicating to a user that said second property sheet system will display said property page selected by said selection means at a position on the display monitor determined by the position of said symbol.

11. A method for displaying property pages on a computer display monitor comprising the steps of

creating a first property sheet system including at least one property page displayable at a first position on a display monitor,
selecting a property page of said first property sheet system, and
creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

12. A method of displaying on a computer display monitor configuration settings governing a peripheral or intercommunication between a computer and a peripheral, comprising the steps of

creating a first property sheet system including at least one property page displayable at a first position on a display monitor,
selecting a property page of said first property sheet system, and
creating a second property sheet system including said selected property page, said selected property page being displayable at a second display position on the display monitor.

13. The method of Claim 11 or Claim 12 further including the steps of

selecting a property page from one of said first and second property sheet systems, and
docking said property page selected from one of said first and second property sheet systems into the other of said first and second property sheet systems for display thereby.

14. A method for displaying property pages on a display monitor, comprising the steps of

creating first and second property sheet systems, said first property sheet system including a property page displaying at first display position on the display monitor,
selecting a property page of said first property sheet system, and
docking said selected property page into said second property sheet system, said selected property page displayable at a second display position on the display monitor.

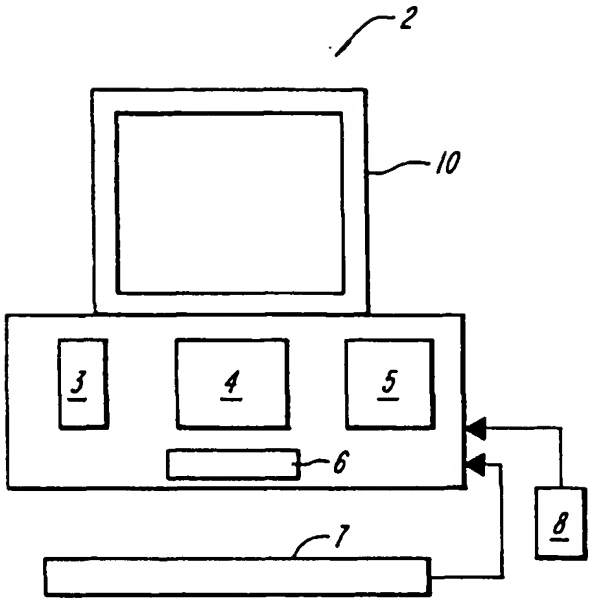


FIG. 1

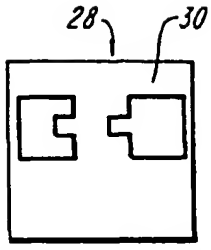


FIG. 5A

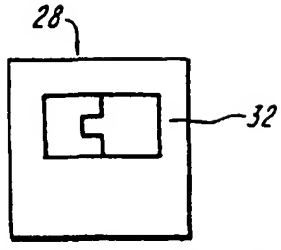


FIG. 5B

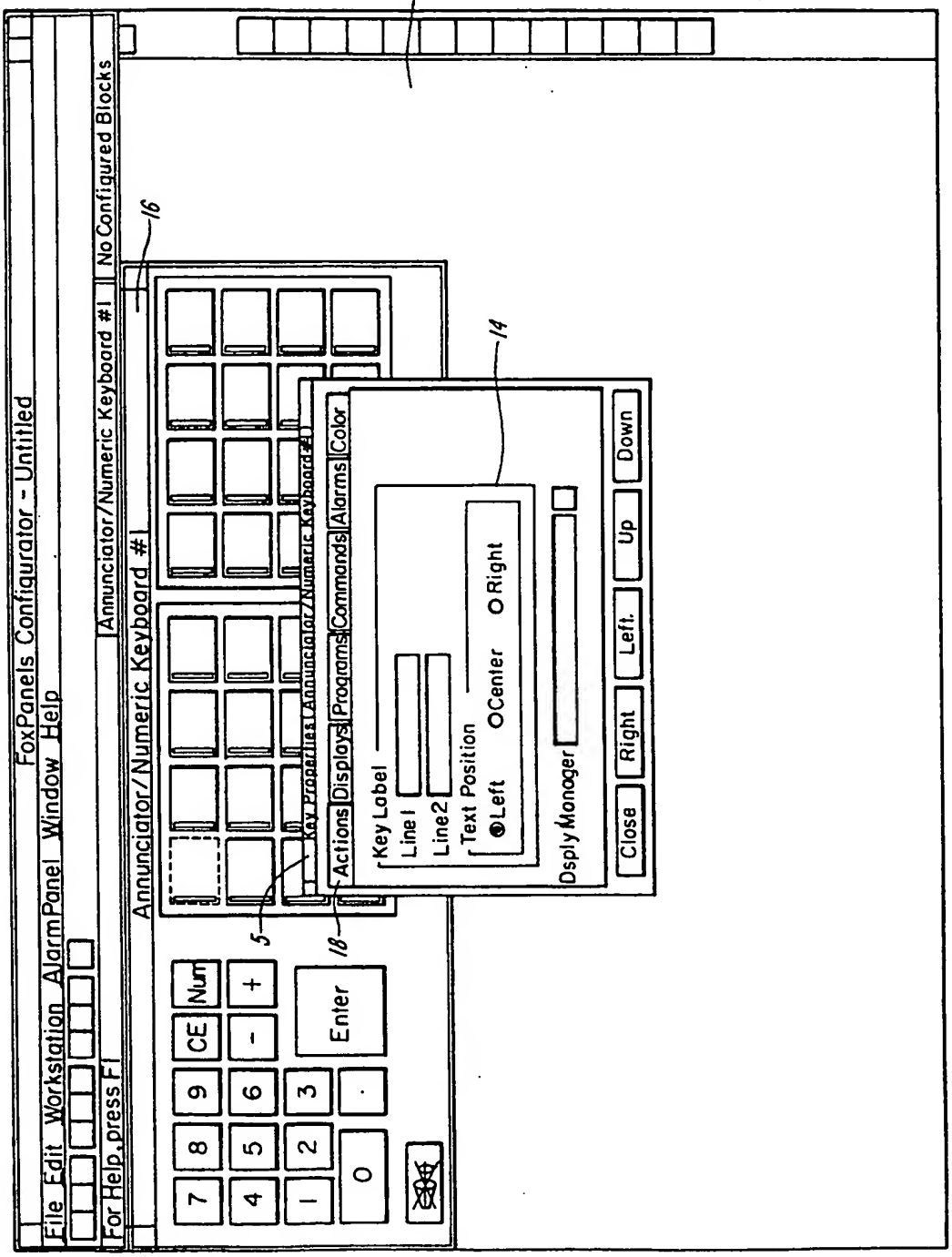
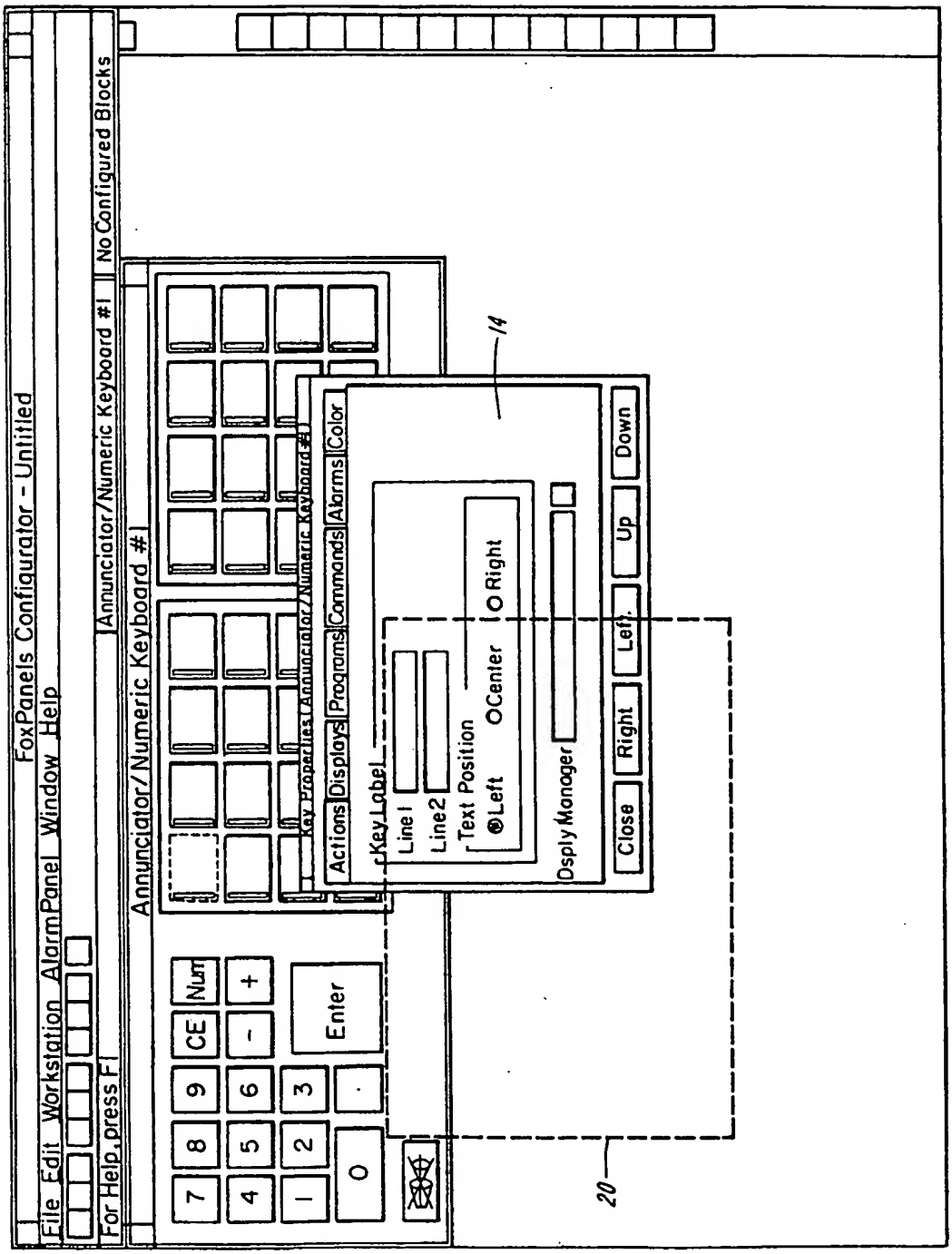

12
13
16
14
5
18
FoxPanels Configurator - Untitled
File Edit Workstation AlarmPanel Window Help
For Help, press F1
Annunciator/Numeric Keyboard #1
No Configured Blocks
Annunciator/Numeric Keyboard #1
Key Properties[Annunciator/Numeric Keyboard #1]
Actions
Displays
Programs
Commands
Alarms
Color
Key Label
Line1
Line2
Text Position
Left
Center
Right
Dsply Manager
Close
Right
Left
Up
Down
7
8
9
CE
Num
4
5
6
-
+
1
2
3
Enter
0
.

FIG. 2

FIG. 3


FoxPanels Configurator - Untitled
File Edit Workstation AlarmPanel Window Help
For Help, press F1
Annunciator/Numeric Keyboard #1
NoConfigured Blocks
Annunciator/Numeric Keyboard #1
Key Properties [Annunciator/Numeric Keyboard #1]
Actions
Displays
Programs
Commands
Alarms
Color
Key Label
Line 1
Line 2
Text Position
Left
Center
Right
Dsply Manager
Close
Right
Left
Up
Down
7
8
9
CE
Num
4
5
6
-
+
1
2
3
Enter
0
.
14
20

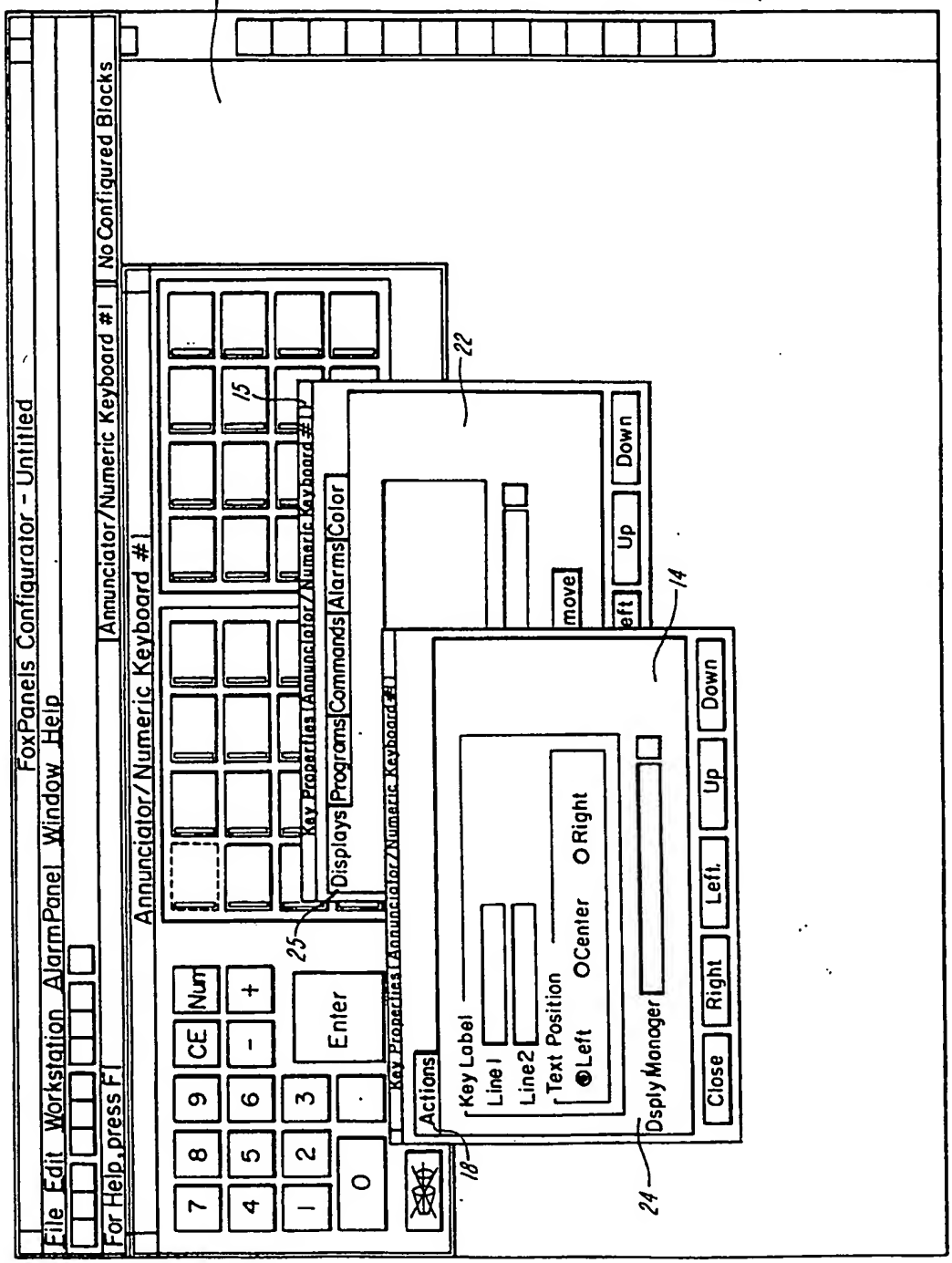

FoxPanels Configurator - Untitled
File Edit Workstation AlarmPanel Window Help
For Help, press F1
Annunciator/Numeric Keyboard #1
No Configured Blocks
Annunciator/Numeric Keyboard #1
Key Properties [Annunciator/Numeric Keyboard #1]
Displays
Programs
Commands
Alarms
Color
Actions
Key Label
Line1
Line2
Text Position
Left
Center
Right
Dsply Manager
Close
Right
Left
Up
Down
move
13
14
15
18
22
24
25

FIG. 4

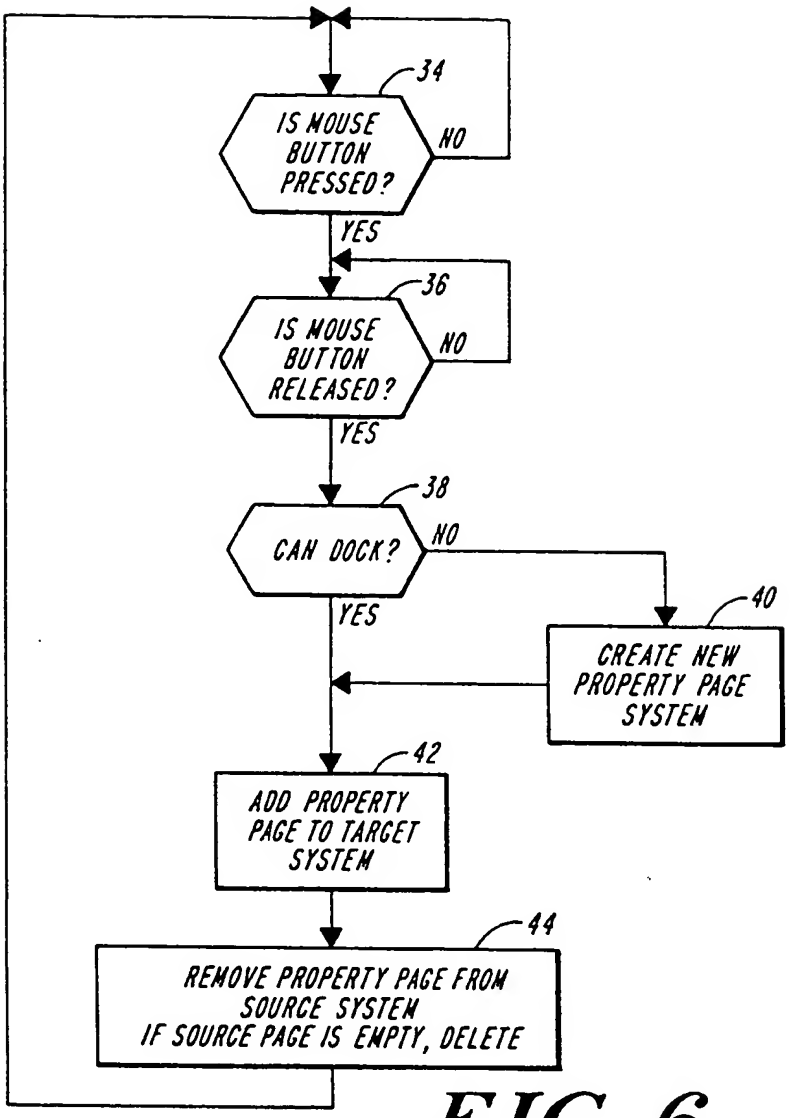

34
IS MOUSE BUTTON PRESSED?
NO
YES
36
IS MOUSE BUTTON RELEASED?
NO
YES
38
CAN DOCK?
NO
YES
40
CREATE NEW PROPERTY PAGE SYSTEM
42
ADD PROPERTY PAGE TO TARGET SYSTEM
44
REMOVE PROPERTY PAGE FROM SOURCE SYSTEM
IF SOURCE PAGE IS EMPTY, DELETE

FIG. 6

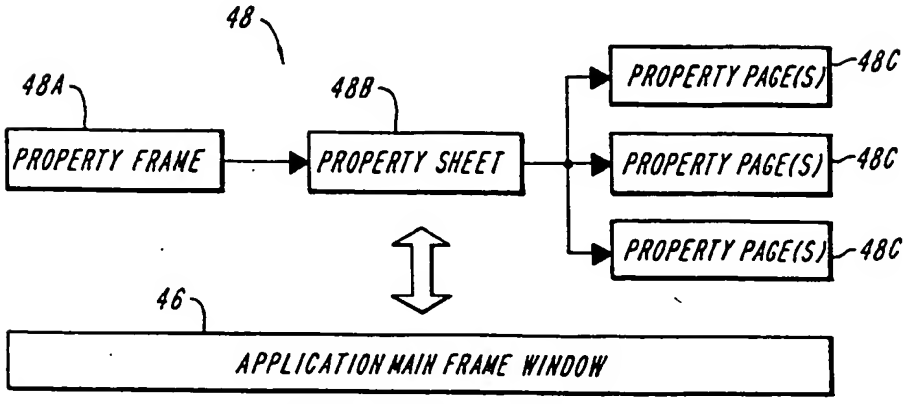

48
48A
PROPERTY FRAME
48B
PROPERTY SHEET
PROPERTY PAGE(S)
48C
PROPERTY PAGE(S)
48C
PROPERTY PAGE(S)
48C
46
APPLICATION MAIN FRAME WINDOW

FIG. 7

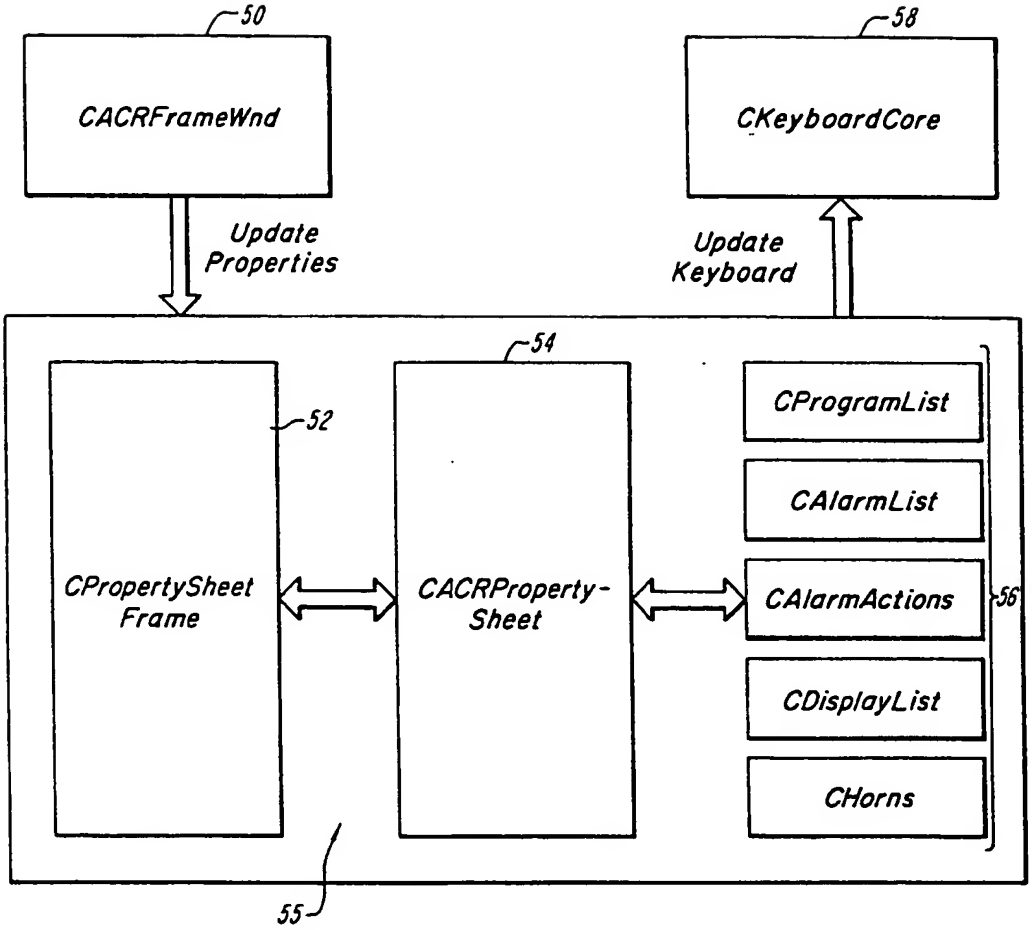

50
CACRFrameWnd
58
CKeyboardCore
Update Properties
Update Keyboard
54
52
CPropertySheet Frame
CACRProperty-Sheet
CProgramList
CAlarmList
CAlarmActions
CDisplayList
CHorns
56
55

FIG. 8

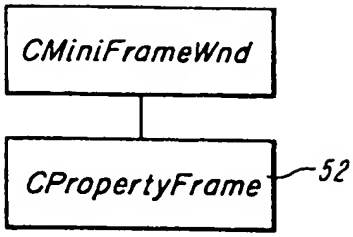


FIG. 9A

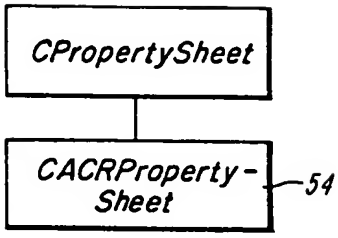


FIG. 9B

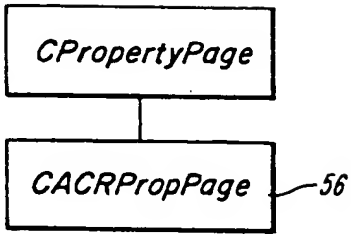


FIG. 9C

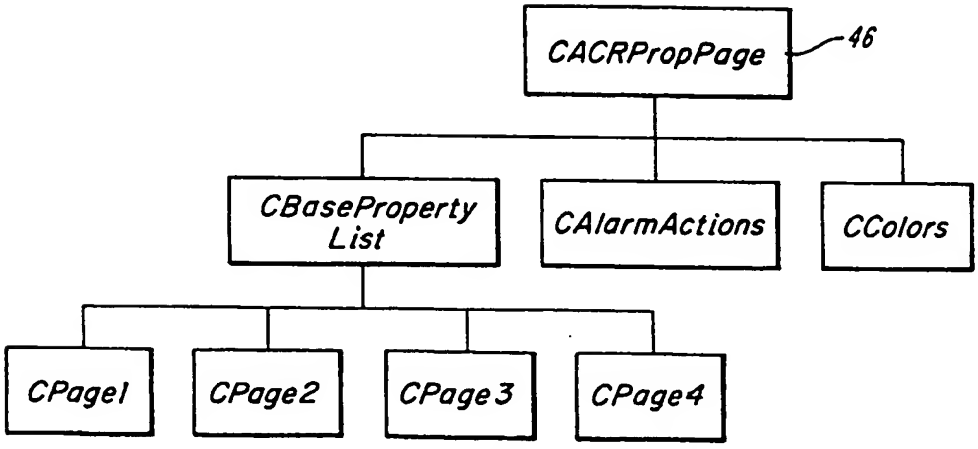


FIG. 10

***This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☑ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☑ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☑ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☑ OTHER: small prints

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.